AWS Elastic Map Reduce Manual

CDT Madeleine M. Schneider

Advisor: Kyle King

AY19-1

Table of Contents

Data Storage in S3…………………………………………………………………………………………………………………………………..3

Regions and Availability Zones………………………………………………………………………………………………………………..3

Choosing you cluster……………………………………………………………………………………………………………………………….3

Activating your cluster…………………………………………………………………………………………………………………………….4

Creating an EC2 Key…………………………………………………………………………………………………………………………………5

Connecting to your Cluster………………………………………………………………………………………………………………………5

Running a spark shell……………………………………………………………………………………………………………………………….6

Viewing Cluster/Job statistics while running……………………………………………………………………………………………6

Viewing Log Files…………………………………………………………………………………………………………………………………….7

**Storage Options**

**S3**

Amazon S3 is [object storage](https://aws.amazon.com/what-is-cloud-object-storage/) built to store and retrieve any amount of data from anywhere – web sites and mobile apps, corporate applications, and data from IoT sensors or devices. It gives customers flexibility in the way they manage data for cost optimization, access control, and compliance.

S3 is easy to use and very cheap. Items are stored in “buckets”. Buckets have typical file structures. Read and write permissions can be set on buckets or folders. Generally, these should not be public!

**For more information on pricing:** <https://aws.amazon.com/s3/pricing/>

**Regions and Availability Zones**

The AWS Cloud spans 55 Availability Zones within 18 geographic Regions and 1 Local Region around the world. Storage and computations cost different amounts depending on the region.

**Choosing your region**

You should choose the region where your data is stored. This is likely the region of your S3 bucket if that is where you are storing data.

**Accessing your S3 bucket**

If your s3 bucket is in the N. Virginia region use the following endpoint: s3.amazonaws.com/bucket\_name

Alternative possibilities:

* s3://bucket\_name (This is the one I used for my project and it worked well)
* [http://s3.amazonaws.com/*bucket\_name*](http://s3.amazonaws.com/bucket_name)
* http://*bucket*.s3.amazonaws.com

Other regions change s3 to s3-region. Ex: -eu-west-1

**For more information:** <https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingBucket.html>

**Choosing your cluster**

**Instances**

Amazon provides a wide selection of instance types optimized to fit different use cases. Instance types comprise varying combinations of CPU, memory, storage, and networking capacity and give you the flexibility to choose the appropriate mix of resources for your applications. Each instance type includes one or more instance sizes, allowing you to scale your resources to the requirements of your target workload.

**For more information on the specific instances:** <https://aws.amazon.com/ec2/instance-types/>

**Demand Instances**

With On-Demand instances you only pay for EC2 instances you use. Once demand machines are purchased they will not be taken away. The use of On-Demand instances frees you from the costs and complexities of planning, purchasing, and maintaining hardware and transforms what are commonly large fixed costs into much smaller variable costs.

**Important Note on Costs**:

* Cost = instance\_cost \* number\_of\_instances\*hours run.
* Don’t blow your whole budget on one run
* Cost is calculated on actual hours used not normalized hours (So don’t freak out if you see a large number)

**For more information regarding demands costs:** <https://aws.amazon.com/ec2/pricing/on-demand/>

**Spot Instance**

Amazon EC2 Spot Instances offer spare compute capacity available in the AWS cloud at steep discounts compared to On-Demand instances. Spot Instances enable you to optimize your costs on the AWS cloud and scale your application's throughput up to 10X for the same budget. Spot Instances can be interrupted by EC2 with two minutes of notification when EC2 needs the capacity back.

**For more information regarding demands costs:** <https://aws.amazon.com/ec2/spot/pricing/>

**Activating your cluster**

**Using the console**

1. Go to the Amazon EMR console
2. Press create cluster
3. In quick settings set
   1. Cluster name
   2. Location of log files (If you keep the recommended location the log files from all created clusters will be organized in same bucket)
   3. Applications (For a scala-spark application choose Spark)
   4. Instance type
   5. EC2 key pair
      1. See *Creating an EC2 Key*
4. If you used advanced settings you may wish to:
   1. Set specific configurations for your machine
      1. Ex: [{"classification":"spark-defaults","properties":{"maximizeResourceAllocation":"true"}}]
      2. With maximizeResourceAllocation = true EMR automatically configures the cluster in the most appropriate way given its hardware configuration
   2. Add steps for the cluster to run once it boots
   3. Configure different instances in same cluster

**Creating an EC2 Key**

Amazon uses public–key cryptography to encrypt and decrypt login information. You will need a key to log in to your instance. You must create a key pair, specify the name of the key pair when you launch the instance, and provide the private key when you connect to the instance.

**Creating a Key Using Amazon EC2**

1. Open the Amazon EC2 console
2. In the Navigation pan, under Network & Security choose key pairs
3. Choose Create Key Pair
4. The private key will be downloaded to your browser
   1. Save to a safe locations – this is the only chance to save the key, if you lose it you will need to make a new one
5. Confirm that the permissions of your private key are set so that only you can read it
   1. Linux/Mac: chmod 400 key\_name.pem
   2. Windows: Use gui, file>properties>security
      1. If this is not set properly you may get an “unprotected private key file” error when trying to connect to an instance, **for more information:** https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/TroubleshootingInstancesConnecting.html#troubleshoot-unprotected-key

**Connecting to your cluster**

It is easy to SSH into the executor or master node of your cluster. Connecting is helpful because it allows you to run code line by line in the spark shell

**How to connect**

1. Start PuTTY.
2. In the Category list, click Session.
3. In the Host Name field, type **the name given when you click the ssh button on the EMR console**
4. In the Category list, expand Connection > SSH, and then click Auth.
5. For Private key file for authentication, click Browse and select the private key file (**key1.ppk**) used to launch the cluster.
6. Click Open.
7. Click yes to dismiss the security alert.

**Note:** See “view cluster/job statistics while running” to connect in a way that enables viewing statistics

**Running a spark-shell**

Using the spark shell is absolute essential for debugging. This is essentially the only way to see print statements in code. Launch the shell with the command *spark-shell*. If you have any dependencies, you’ll want to get the information from your scala build file. The best way to do this is by assembling your application as a jar file and putting that file in s3. Then run the command *spark-shell --jars s3://bucket/folder/name\_of\_jar.jar*

**Note:** Spark shells instantiates a spark session as *spark* and its context as *sc*. Thus, you should not create these when you are running lines of code in the shell

**Viewing cluster/job statistics while running**

**First time:**

1. Download foxy proxy Chrome extension and restart Chrome
   1. <https://chrome.google.com/webstore/detail/foxyproxy-standard/gcknhkkoolaabfmlnjonogaaifnjlfnp?hl=en>
2. On the cluster terminal summary page, next to Connections, click *Enable Web Connection*
3. Copy and paste the information under step 2 part 3 into a text editor. Save as foxy-proxy-settings.xml
4. Click on the FoxyProxy icon in the toolbar and select options
5. Click Import/Export
6. Choose the foxy-proxy-settings.xml file
7. Click add
8. Click on the FoxyProxy icon and choose the setting “Use proxies based on their pre-defined patterns and priorities”

**Consecutive times:**

1. Configure putty sections as if connecting normally
2. On cluster terminal summary page, next to Connections, click *Enable Web Connections*
3. Under Step 1 part 8 see bolded number
4. In Putty Click SSH>Auth>Tunnels
5. Put the bolded number from step 11 into the source port field
6. Select Dynamic and Auto settings
7. Click open
8. Now on the cluster terminal summary page, next to connections, you will see options to view the status of a job and of the cluster

**Viewing Log Files**

Note: Log files may take time to populate

If you did not change the default output location of log files you can find in your S3 bucket at:

aws-logs… > elasticmapreduce > cluster-id (found on cluster terminal summary page)

If you are trying to determine why a step or application failed go to:

Containers > application-number (found on cluster terminal application history page)

Log files give an overwhelming amount of information. They also may not be immensely useful given the distributed nature of EMR.